SE 3354 – Software Engineering Project

Created a solution (campus communication app) under a compact software engineering process. It started with creating a team and assigning a project manager for the team. Every two to four weeks the project manager changed to ensure everyone got a chance to experience the responsibilities. The project manager was responsible for keeping the meeting logs, all the assignments assigned to team members, member deliverables, and reporting meeting minutes. The first task was creating a Statement of Work (SOW) which includes purpose of work, purpose of document, functional and non-functional requirements, and more. I worked on functional requirements, acceptance criteria, pricing and agreement. Next, was the software engineering document which included business landscape, introduction of document, requirements, design, coding, testing, deployment, maintenance. We started creating diagrams for our solution with me doing the sequence diagram and context diagram. Next, we worked on use cases for our solution. In addition, I had to work on the business landscape and document titles. In the document I had to describe what technologies we were going to use from the backend to the front-end. I had to work on the testing process and all the phases with it. I had regression testing with sub testing, scenario testing and performance testing, alpha and beta testing, and acceptance testing. Then we had to create test cases for our functional requirements. Then I had to describe the maintenance and how the transition was going to be about between the developers to the maintenance team. Finally, we concluded by doing the wireframes.

Created a campus communication app under a compact software engineering process. No coding was done but rather the goal of project was to understand the software engineering process that included the business landscape, requirements, design, the coding phase, testing, deployment, and maintenance of a project. First, as a team we decided an agile approach would be best suited for our project. Next, the areas that I worked on were acceptance criteria, agreement, functional requirements, sequence diagrams, and context diagrams to further understand our solution. In the coding phase, I specified the back-end technologies and in the testing, phase specified our process that included test cases. Finally, I specified the transition of the solution from developers to the maintenance team.

Under the project I had the chance to work on the acceptance criteria, pricing, agreement, and functional requirements. In the design phase I created sequence diagrams and context diagrams to understand the functionalities of our solution and what systems will it be interacting with. In the coding phase, I specified what back-end technologies we will be using and along with summarizing our technology stack in the document. In the testing phase, I specified our process which started with regression testing after each sprint and concluded with acceptance testing. Afterwards, we created test cases for both the functional and non-functional requirements specified earlier. Finally, it concluded with me specifying the transition between developers to the maintenance team.

SE 4347 – Database Projects

Created a database using SQL for a library that has the following modules: person, employee, member, books, borrow details and payment. The specifications and constraints of the database were written before. As a team, we first drew an EER diagram to have a conceptual design of the database and accurately represent the set of requirements. After, the EER diagram was converted into a logical design then to database design which we then normalized all tables to the third normal form. In addition, dependency diagrams were created for each table. Finally, SQL statements were created to create the database along with the specifications and constraints followed in our database design. As well as SQL statements for views specified by the client and any other queries needed in the problem description. All queries followed the constraints in the description and database design.

Created a database using SQL for a library that has the following modules: person, employee, member, books, borrow details and payment. As a team, we first drew an EER diagram to have a conceptual design of the database. After, the EER diagram was converted into a logical design then to database design which we then normalized all tables to the third normal form. In addition, dependency diagrams were created for each table. Finally, SQL statements were created to create the database along with the specifications and constraints followed in our database design. As well as SQL statements for views specified by the client and any other queries needed in the problem description. All queries followed the constraints in the description and database design.

SE 4348 – Operating Systems

Project 1/2: Concurrent Prefix Sum

Designed a concurrent program in C++ in a Linux system that computes the prefix sum of a given array in a file and exports the sum into an output file. This program was divided into two separate and independent programs to demonstrate the difference between the concurrency of threads and processes. The user was allowed to specify the number of threads and processes and algorithm accounted for any uneven amount of work to be done and threads/processes created. The processes were created using fork and it used my very own modification of an algorithm. The threads were created by the new thread function along with it, the boost barrier was used for concurrency control and ensure no overlapping critical sections.

Project 3: Unisex Bathroom Problem

Created a Java solution using semaphores to the unisex bathroom problem in which the user specifies the number of toilets available in a restroom. The problem is that men can enter if toilets are available and same goes for women, but one gender cannot enter if the opposite gender is in the restroom. The solution was done using threads and semaphores. The solution is also deadlock free so the threads should concurrently be doing work dependent on the semaphore.

SE 3340 – Computer Architecture

MIPS Programming Basics:

Created a simple MIPS program that uses the registers for three input values and three output values. In addition, a variable is used to hold the user’s name and three variables are used to hold messages. The program asks the user for its name which is saved to memory and for three integers. Then simple calculations are done on the three input integers which are then displayed to the user.

MIPS Control Structures:

Created a MIPS program that uses system calls to get an input string from the user and the program counts the amount of words and characters in the words. This data is then returned from a function in specified registers. The results are outputted to the user and repeated until the user enters a blank string or cancels. The stack registers are used to keep track of the data. Finally, another system call is used to say goodbye before the program ends.

BMI Calculator:

Created a MIPS program that calculates the BMI of the user that inputs its name, height, and weight. The program then stores the data into registers, does calculations in the registers, and stores the BMI into another register. The program then outputs the BMI to the user along with his/her name.

MIPS FP Operations:

Created a MIPS program that reads an input file into a buffer in memory. The string “numbers” are extracted from the file and converted into integers. The integers are then stored in an array. The array is then printed to the console. The array is then sorted using selection sort and followed by another print to the console. Finally, the program calculates the mean, median, and standard deviation followed by another print to the console.

Overall:

Created multiple MIPS projects ranging from simple use of registers for input data of the users and output to the console, to using system calls to get the input from the users. Simple calculations were also done to the registers along with the use of other system calls to display a dialogue box. Functions, loops, conversions, and more were used for many programs such as calculating a BMI. Finally, buffers in memory were used to read from input files and many algorithms were used in MIPS such as selection sort. Please see GitHub for more specific details.

SE 3345 – Data Structures

Sieve of Erotosthenes Algorithm:

Created a Java program implementing the Sieve of Erotosthenes algorithm to compute all prime numbers less than equal to a given integer N.

Singly Linked List Inventory:

Created a Java program using a singly linked list of a specific generic type for an inventory that has a price, name of item, id, and item description. This generic singly linked list Java class will hold objects of the generic type <AnyType> and will extend an interface. In addition, the class will have its own set of functions that include making the linked list empty, finding a specific node, inserting a node at the front, deleting from the front, regular delete, and print total amount of nodes. The program will read from file that has many commands and the program will read the commands and execute the commands. The program will output a file with any necessary information.

Lazy Binary Search Tree

Created a Java program that creates a binary search tree with lazy deletion. A class is created for the tree node and another for the lazy binary search tree. The lazy binary search tree has the following functions (methods): insert (throws IllegalArgumentException) if the id of the node is invalid, delete (throws IllegalArgumentException), findMin, findMax, containsID (throws IllegalArgumentException), toString, height, and size. The program will read an input file with certain commands and the program will recognize the commands and create the binary search tree. In return the program will produce an output file writing a response for each command.

Hash Table

Created a Java program that implements a Java hash table structure using linear probing collision strategy. A generic class is used called HashLinearProbe <K, V>, where K is the key and V is the value. This class will have a private static class HashEntry <K,V> which will be used to create an array to represent the hashtable: HashEntry <K,V> hashtable[]; The class will have the following functions, insert, find, delete, getHashValue, and rehash.

QuickSort

Create a Java program that implements several variations of the Quick Sort algorithm, each with a different choice of pivot. The program sorts any array using in-place quick sort and the following four different pivot selection strategies: first element, random element, median of three random elements, median of first, center, and last element. A function will then calculate the time to sort the array and an output file (report file) will created inserting information of the array and the times for each quick sort depending on the pivot selection.

Overall: Created multiple Java programs that range from data structures and algorithms. Created Java programs that used the Sieve of Erotosthenes algorithm and the Quick Sort algorithm to using data structures like the linked lists, binary search trees, and hash tables. Please see the GitHub for more details of each project.

SE 3377 – Linux Systems

Fibonacci Sequence:

Created a program in C in a Linux system that perform two different functions via different threads. One of the functions will take a positive integer N, as the parameter and find the sum of these first N consecutive integers. The other function will also take a positive integer N, but will calculate and store an array of integers following the Fibonacci sequence of length N. In main, the program will first ask the user for a greater than zero integer before creating the threads. If the user enters a zero or lower, then an error message will appear, and the user will have to insert another number again until the number is valid. Once a valid number is obtained, the threads will be created and a message indicating the creation of the 2 threads will appear. One of the threads will execute the sum of N integers, while the other thread will calculate the Fibonacci sequence. Once both threads are finished a message will appear indicating they are finished followed by main outputting the sum of the N and the output of the Fibonacci sequence.

Created a program in C in a Linux system that perform two different functions via different threads. One of the functions will take a positive integer N, as the parameter and find the sum of these first N consecutive integers. The other function will also take a positive integer N, but will calculate and store an array of integers following the Fibonacci sequence of length N. Once both threads are finished a message will appear indicating they are finished followed by main outputting the sum of the N and the output of the Fibonacci sequence.

C++:

Created numerous C++ projects throughout my years at my university. Ranging from learning how to code in C++ by using simple variables to more complex programs that are object oriented and use polymorphism. In addition, many data structures have been used throughout the C++ projects such as linked lists, trees, and more. See GitHub for more details.

JP Morgan Code for Good

Developed a web app using HTML, CSS, and JavaScript for a non-profit organization called Canine Assistants. The purpose of the web app was for the dog to communicate with its owner and for help if needed. Me and my team decided to have binary questions displayed based off of two images. As the dog touches the screen visual, auditory, and text alerts are sent to the device and external devices. For example, one image, if clicked, calls a specific number through Skype to alert other people that the dog’s owner is in danger. However, other images can be meant for the necessities of the dog.

Events App

Programmed a mobile app in Java for users to register events that could be intended for students, student organizations, professionals, etc. This would be visually displayed to the people you are connected with. In addition, Firebase was used to make a user login using their credentials using Firebase’s Authentication and to store the information of the events. This data was stored in the Realtime database of Firebase and could be retrieved at any time.

CS1336

Assignment 1: Basic Programming

Used basic techniques in C++ to display content using cout, endl, and escape sequences to print to the console. In addition, basic storage of data in variables was used to calculate the distance based off a formula using acceleration and time. Simple commands were used to fix the output of the console and align them for visual appeal.

Assignment 2: Basic Calculations

Using basic techniques in C++ to convert Celsius to Fahrenheit and Kelvin. The program asks for input in Celsius

Student Organization App

Currently leading a team of 5 into the Agile process of creating a social media type app for student organizations on campuses. The app will include a great deal of options for students to explore and organizers to be prepared. So far, I have created a “Create Event” page for organizers. This page will accumulate any necessary information and store it in the database (Firebase). The page functions like any other form with input validation, error messages, and when completed shows a completion message and redirects it to the home page. My next task is to create a page for users to be able to visually see the event information. This will include retrieving information from the database.

Nova

Currently working on a console based personal assistant to help me record my workouts, my goals, and give me daily briefings of what I should do that current day. The program will respond to specific input which has been predefined and limited to very few actions and perform the necessary functions corresponding to that action requested. So far, I am working on storing the information of the user locally so whenever a user tries to run the program again it will have the information of the user and a daily motivational message should appear with the menu of actions.

Unreal Engine Game

Created a series of three games in Unreal Engine using C++. The first game is a character that needs to pick up objects before its health drains down. If that occurs the player dies and respawns. The movement and entirely of the game were done solely in C++. The next game is breakthrough in which the ball moves and breaks bricks. Again, the entirely of the game was done solely in C++. Finally, the last game is Pac-Man. This game has the basic functionalities (enemies, respawning, teleporting on the edge of the map) of the original game and the project was done using C++ (70%) and the blueprints of Unreal Engine.

Personal Website

Modified existing code from a template provided by Rashmi Ap. The design and UI remained the same. The only modification done to this website is in the “About Page”. The three cards at the bottom previously redirected the user to one of the pages on top menu. However, that was changed to have a pop-up window instead to demonstrate either my basic info, skills, and interests. In addition, the skills pop up has a star rating with each skill and when the window is too small the star rating disappears. Future plans are to add a filter option to the projects in order to filter by language, date, etc.

Instagram Project

Created a program that reads in two files, one that should contain the names of your followers in social media and another that contains the names of the people you are following. The program then uses an algorithm to distinguish which names you are not following and the names of people you don’t follow you. This is then visually displayed to the user.

Previous Personal Website

Developed a react app from scratch as my personal website. The website has a top menu navigation and a side bar menu (if the window is too small) and dynamically displayed data about me. However due to the styling of the website I changed the website for a professional website. In addition, a previous attempt to creating a personal website was done in simple HTML, CSS, and JS but the felt the website was too clustered.

GameZone Review App

Created a react native app that allows a user to create a review, give it a star rating shown in the form of hearts (life), and post it to a feed. The home page is the home feed and there is a add button to display a modal. The modal then uses another component to display and store the state of the review. The review uses Formik and Yup (packages) that enables on submit functions and validation. Then after the review as been submitted, the modal closes and the home page is updated in its feed with the new review.

Todo App (React Native)

Developed a to-do app using react native. The react native app simply uses state and components to update the state shown in the home screen. As the user types in the insert box and clicks submit the state is updated and which updates the visuals in the home feed. In addition, when the to-do is clicked, the state is once more updated by deleting it and so is the visuals. If no to-dos are displayed, then a message saying “No to-dos” is shown to the user.

Todo App (React)

Created a to-do app using react. The to-do provides basic functionality of a to-do app, as the user can add to-dos which is shown in the home feed and it can be deleted from the home-feed. This is done by using the state of the component as it adds and delete to-dos the state is updated. Materialized CSS was used to style to to-do app.

First React Project

Developed a simple react app that enables you to add users to the list of friends. The form is on the home page and dynamically updates your home feed of list of friends. Each component that corresponds to an added friend, a button shows that enables the user to delete that friend. If deleted, the state is updated, and the feed is no longer shows that friend.

SHPE Mobile App

Created a Java mobile app with a team for the student organization called Society of Hispanic Professional Engineers at UTD. The UI was done by me using Adobe XD and after I integrated the UI into Android Studio. In addition, I used Google Authentication provided by Firebase to log in users into the application. The project was soon discontinued due to complications and became the now react native project called Student Organization App.

Uber App

Created a Java mobile app that used Android Studio, GeoFire API, Google Maps API, and Firebase to successfully have a Uber-like application. I used Google Authentication to sign up and log in users and drivers. The users were allowed to request for rides which I used GeoFire API to acquire the coordinates of the locations being requested and Google Maps API to be able to use Google Maps within my application. The requests were saved in the database along with its coordinates. On the driver side the driver could accept the request and see the location of the user. The driver’s location was then periodically updated and saved to the database. Both the users and driver are able to see each other’s location. The user then gets a notification when the driver arrives, and the user is able to cancel a ride at any time.

SHPE Website

Briefly developed a page for the SHPE at UTD website using PHP, HTML, CSS, and JS. The page was intended for members to be able to pay their dues. The page was created using HTML, CSS, and JS, and PHP was then used to connect the page to the server. In addition, I created manually SQL tables in the database for further expansion of our database schema. The page that I worked on was never merged to the main branch as the goals deviated from what the project manager wanted.